Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 4-5 November 2002

A GAME-THEORETIC LEARNING MODEL IN MULTI-AGENT SYSTEMS

CHI ZHANG [a], XIA ZHANG [b], JIAO-LONG WEI [a], MAN-LI ZHOU [a]

[a] Dept. of Electron. and Info., Huazhong University of Science and Technology, 430074 Wuhan, China.
[b] Institute of Fundamental Electronics, University of Paris Sud (Paris XI), 91405 Orsay Cedex, France.
E-MAIL: zhangfx@mail.whut.edu.cn

Abstract:

This paper investigates the problem of learning in multi-agent system that can be applied to tele- communication networks. We model the strategic inter- dependence situation and learning dynamics of self- interested agents in the framework of Markov game with incomplete information. By combining fictitious play's best response strategy and Nash Q-learning's multi-agent Q-learning, we propose a new multi-agent learning algorithm that can maximize learning agent's expected reward and optimize system-wide performance. We also summarize other algorithms from the game theory and reinforcement learning communities, and compare these algorithms with ours.

Keywords:
Multi-agent systems; Multi-agent Q-learning; Markov games; Non-cooperative game; Reinforcement learning

1 Introduction

In recent years, certain aspects of the multi-agent systems and machine learning communities have merged to create a new field of wide interest: multi-agent learning [1-2]. Research in this area is concerned with learning not in the classic case of an agent and its environment, but rather with interactive learning among multiple agents. My research is focused on learning problem of a certain kind of multi-agent system (MAS) which is suited to be applied in telecommunication networks, such as bidding in on-line auctions and dynamic pricing on bandwidth in network contexts [3].

In network contexts, because of distance, population size, or selfishness, agent cannot or will not make effort to achieve the best rewards of other agents. The goal of a self-interested agent within a MAS is to maximize its own expected reward over time. In a situation of strategic interdependence, where the actions of one agent may affect the rewards of other agents, the optimal behavior of an agent must be conditioned on the expected behaviors of the other agents in the system. Game theory [4] is a useful tool because it predicts the strategies that rational agents will choose to play in a particular game.

In standard game theory, agents use complete information to find equilibrium and make decision following equilibrium strategy. Because the information available to agent is often limited, and the essential problem in game's equilibrium, agents need to learn what is the best strategy through trial and errors. Reinforcement learning techniques have addressed this problem for a single agent acting in a stationary environment, which is modeled as a Markov decision process (MDP). But, multi-agent environments are inherently non-stationary since the behavior of other agents may change as they also learn and adapt and the presence of other learning agents is outside of our control.

To solve those problems, we need a new model to describe self-interested agents interacting with each other and their interactions change over time with incomplete information. Markov game [5], as a natural extension of MDP to include multiple agents, is a suitable mathematic tool to model this dynamic interaction situation. A new learning algorithm is also needed to help learn agent to find the solution of this Markov game.

This paper is organized as follows. Section 2 analyzes the background of designing multi-agent learning algorithms, and summarizes the requirements of algorithms in network context. Section 3 provides the theoretical framework for Markov game. To solve this game, in Section 4 we present a new algorithm called Multi-agent Q-learning with best response strategy. Section 5 summarizes other algorithms from the game theory and reinforcement learning communities, and compares these algorithms with ours. Section 6 concludes with a brief summary and discussions of the future work in multi-agent learning.

2 Background

2.1 Network context

It would be foolish to claim that a certain learning method is the best for all MASs. Everything depends on the details of the context and the designing of learning algorithms is not "one size fits all". In this paper we focus our attention on learning problem that occurs in what we call a network context. In network contexts, agents interact through the common use of a resource, such as a communication link or a shared database, which is accessed over a network. The interactions of Internet congestion control algorithms where agents share network bandwidth, as we described in [3], are perhaps the most studied

0-7803-7508-4/02/$17.00 ©2002 IEEE

examples of a game in a network context. As the Internet continues to grow, and more resources are shared by remote users, we expect the network context to become increasingly common. The context of network games differs from the other contexts in five important ways [6], which are the essential properties of corresponding multi-agent systems.

Non-cooperative games: In non-cooperative games, agents pursue their individual goals and make decisions independently. In most cases, having a central authority controlling the overall network is unrealistic.

Limited information: Agents have extremely limited information pertaining to the characteristics of the shared resource; in other words, they do not know the underlying structure of the game. Moreover, agents are not explicitly aware of the existence of other agents, as there is no way of directly observing the presence of others, and they are therefore incapable of accurately modeling their opponents.

Dynamic structure: The structure of the game, in terms of agents, strategies, and rewards, all subject to change over time. Shared resources like network links and servers periodically crash, often experience other unpredictable changes in their capabilities, such as upgrades or route changes. Moreover, users of network resources come and go frequently.

Automation: Play is often carried out by computer algorithms, rather than by human users. For instance, congestion control algorithms (e.g., TCP), which are embedded in operating systems, manage the sharing of network links. So it is reasonable to assume that agents' strategies are stationary.

Asynchrony: Games are played in an asynchronous fashion, without any notion of definable rounds of play, since the rates at which agents adapt their strategies vary widely. It is difficult for agent to make optimal decision. So agents have rational limitations in different degrees.

2.2 Criteria of learning algorithms

In network contexts, the learning algorithm for the agent when in the presence of other learning agents need to satisfy the following criteria:

Online-learning: An online learning agent needs to care about her run-time performance at any time. There is no separate training period for testing actions randomly. An agent has to choose action carefully at each step to maintain a reasonable performance. A learning algorithm in network contexts has to be online because agents constantly interact in a dynamic and normally there is no free trial.

Rationality: If the other agents' policies converge to stationary policies then the learning algorithm will converge to a policy that is a best response to their policies. This requests the learning algorithm takes account other agent's actual strategy. In network contexts, because the agents are heterogonous, network game may have multiple equilibria and opponent may not be playing optimally. So, learning independently is not possible.

Scalability: Every algorithm in network must be scalable, because the number of agent and state is enormous and still increasing.

3 Theoretical framework

The engineering of MASs composed of learning agents brings together techniques from Markov decision processes (MDPs) and non-cooperative games (NGs). In this section, we first provide a concise review of the necessary concepts in these two fields. Then we setup the framework for Markov games, which can be seen as the merging of MDPs and NGs.

3.1 Markov decision processes

A discrete-time stochastic process is a tuple, $\langle T, S, A, P, r \rangle$, where time is discrete: i.e., $t \in T = \{0,1,\cdots\}$, S is a (finite) set of states ($s \in S$), A is a (finite) set of actions ($a \in A$), P is a probability transition function that describes transitions between states, conditioned on past states and actions: e.g., $P[s_{t+1} \mid s_t, \cdots, s_0, a_t, \cdots, a_0]$, and r is a reward function.

A Markov decision process is a stochastic process whose probability transition function and reward function satisfy the Markov properties:

$$P[s_{t+1} \mid s_t, \cdots, s_0, a_t, \cdots, a_0] = P[s_{t+1} \mid s_t, a_t] \quad (1)$$

$$r_t = r(s_t, a_t) \quad (2)$$

In words, r_t and s_{t+1} depend only on current state and action. A policy is a map from states to actions: i.e., $\pi(s): S \to A$. Solving MDPs consists of finding a policy π^*, which determines the agent's actions so as to maximize discounted future reward, with discount factor γ ($0 \le \gamma \le 1$). When the agent does not know the reward function or the state transition probabilities, reinforcement learning is needed for the agent to associate his actions with the rewards by interacting with the environment through trial and errors. Q-learning which is proposed by Watkins and Dayan [7] in 1992 established the connection between reinforcement learning and MDPs. In this learning approach, the agent can directly learn about his optimal policy even without knowing the state transition probabilities. We define:

$$Q(s,a) = r(s,a) + \gamma \sum_{s' \in S} P[s' \mid s,a] V^*(s') \quad (3)$$

$$V^*(s) = \max_{a \in A} Q(s,a) \quad (4)$$

In words, the Q-value function $Q(s,a)$ is the immediate reward obtained at state s for taking action a plus the expected discounted value of the future rewards obtained by following the optimal policy thereafter. The state-value function $V^*(s)$ at state s is defined as the value that maximizes $Q(s,a)$ over all actions a. The actions that

maximize $Q(s,a)$ at each state s describe the optimal policy π^*: i.e.,

$$\pi^*(s) \in \arg\max_{a \in A} Q(s,a) \quad (5)$$

Let $\hat{Q}$ denote learner's current approximation to Q. The learning rule is: observe current state s, select an action a and execute it, then receive immediate reward r; observe the new state s' and update the table entry for $\hat{Q}(s,a)$ as follows:

$$\hat{Q}_t(s,a) \leftarrow (1-\alpha_t)\,\hat{Q}_{t-1}(s,a) + \alpha_t[r + \gamma \max_{a'} \hat{Q}_{t-1}(s',a')] \quad (6)$$

where α_t is the learning rate and decays over time.

3.2 Non-cooperative games

A non-cooperative game in normal form is a tuple $\langle N,(A_i),(r_i)\rangle$, where $N=\{1,\cdots,n\}$ is the finite set of players (agents), A_i is the action space for agent i ($i \in N$), and r_i is payoff function that gives agent i 's reward $r_i(\vec{a})$ for each actions profile $\vec{a}=(a_1,\cdots,a_n)$, where $a_i \in A_i$. Agent i 's strategy σ_i is a probability distribution over action space A_i. Each agent's random- ization is statistically independent of those of other agents. We denote the space of agent i 's strategies by $PD(A_i)$, where $\sigma_i(a_i)$ is the probability that σ_i assigns to a_i. The space of strategy profiles is denoted by $\times_{i \in N} PD(A_i)$, with element $\vec{\sigma}=(\sigma_i,\cdots,\sigma_n)$. Then the agent i 's expected reward from strategy profile $\vec{\sigma}$ is:

$$r_i(\vec{\sigma}) = \sum_{\vec{a} \in A} \{[\prod_{j=1}^{n} \sigma_j(a_j)]\, r_i(\vec{a})\} \quad (7)$$

Definition 1 Given other agents' strategies $\vec{\sigma}_{-i} = (\sigma_1,\cdots,\sigma_{i-1},\sigma_{i+1},\cdots,\sigma_n)$, strategy $BR_i(\vec{\sigma}_{-i})$ is a best-response (BR) for agent i ,iff

$$BR_i(\vec{\sigma}_{-i}) = \arg\max_{\sigma_i \in PD(A_i)} r_i(\sigma_i,\vec{\sigma}_{-i}) \quad (8)$$

Definition 2 An agent is said to be rational, if he always plays a best response.

Definition 3 A Nash equilibrium is a strategy profile $\vec{\sigma}=(\sigma_i,\cdots,\sigma_n)$, with $\sigma_i \in BR_i(\vec{\sigma}_{-i})$, for any $i \in N$.

If agent i believes that other agents are rational, the best response strategy and Nash equilibrium strategy are the same for agent i.

3.3 Markov games

Stochastic games generalize NGs and MDPs. A stochastic game is a tuple $\langle T,N,S,(A^i),P,(r^i)\rangle$, where N is a set of n players, S is a set of states, A^i is the i th agent's set of actions, P is a probability transition function that describes state transitions, conditioned on past states and joint actions, and $r^i(s,\vec{a})$ is the i th player's reward for state $s \in S$ and joint actions $\vec{a} \in A(s) = A^1(s) \times \cdots \times A^n(s)$. Stochastic games for which the probability transitions and reward functions satisfy the Markov property are called Markov games: i.e., for $\vec{a}_t = (a_t^1,\cdots,a_t^n)$,

$$P[s_{t+1} \mid s_t,\vec{a}_t,\cdots,s_0,\vec{a}_0] = P[s_{t+1} \mid s_t,\vec{a}_t] \quad (9)$$

$$r_t^i = r^i(s_t,\vec{a}_t) \quad (10)$$

A Markov game is a NG, if it has only one state; a MDP if it has only one player (agent). So we can say that Markov games contain both MDPs and NGs as subsets of the framework.

In the paper we focus on the learning behavior of a single agent, and seek to find the best policy for that agent, but the environment includes multi-agents. Unlike MDPs, there is not likely an optimal solution to a Markov game that is independent of the other agents. This makes solving MGs for a single agent difficult to define. The existence of equilibria does not alleviate our problem since it is only equilibrium if all agents are playing the equilibrium. There may also be multiple equilibria, so to which equilibria play is dependent on the other agents.

In MGs with incomplete information, agents do not know other agents' (and even their own) reward functions. We assume that the agent can observe the immediate reward of his own at each time period, and the previous actions of all the agents are observable. Using learning algorithm in MAS, the agent can gradually learn the optimal action for each state.

4 Multi-agent Q-learning with best response strategy

4.1 Q-value in Markov games

An agent's policy is a plan for playing a game. Agent i 's policy $(\pi_0^i,\cdots,\pi_t^i,\cdots)$ is defined over the entire course of the game, where π_t^i is called the decision rule (or strategy) at time t. If $\pi_t^i = \pi^i$ for all t, agent i 's policy is called stationary strategy, that is, the decision rule is independent of time. Agent i 's strategy $\pi^i(s) : S \to PD(A^i)$ is a probability distribution over action space A^i for state s. We denote $\pi^i(s,a^i)$ is the probability that $\pi^i(s)$ assigns to $a^i \in A^i$. The space of strategy profiles is denoted by $\times_{i \in N} PD(A_i)$, with element $\vec{\pi}(s) = (\pi^1(s),\cdots,\pi^n(s))$.

In Markov games, agent i 's optimal Q-values are defined over states and action-vectors, rather than state-action pairs (see Eq. (3)):

$$Q^i(s,\vec{a}) = R^i(s,\vec{a}) + \gamma \sum_{s' \in S} P[s' \mid s,\vec{a}]V^{i*}(s') \quad (11)$$

And the general multi-agent Q-learning algorithm is:

$$Q_{t+1}^i(s,\vec{a}) \leftarrow (1-\alpha_t)Q_t^i(s,\vec{a}) + \alpha_t[r_t^i + \gamma\, V^{i*}(s')] \quad (12)$$

Let $Q^i = \{Q^i(s) \mid s \in S\}$ be agent i's Q-table and $Q^i(s)$ the Q-table under state s with each element represented by

$Q^i(s,\vec{a})$.

The definition of the state-value function $V^{i*}(s)$ in Markov game depends on the strategy of learning. In the paper, we propose a new learning algorithm with the goal to learn Q-value with best response strategy.

Each state in a Markov game can be viewed as a NG (called stage NG). Consider a stage NG at time t. Let $Q_t^i(s,\vec{a})$ be the reward of agent i for actions profile $\vec{a}=(a^1,\cdots,a^n)$ and state s. Then the agent i 's expected reward from strategy profile $\vec{\pi}(s)$ is:

$$Q_t^i(s,\vec{\pi}(s)) = \sum_{\vec{a}\in A}\{[\prod_{j=1}^{n}\pi^j(s,a^j)]\,Q_t^i(s,\vec{a})\} \quad (13)$$

Given other agents' strategies $\vec{\pi}^{-i}(s)$ $=(\pi^1(s),\cdots,\pi^{i-1}(s),\pi^{i+1}(s),\cdots,\pi^n(s))$, we suggest that agent i adopt a best response strategy to update her Q-value and define it as the best response Q-value, $\mathrm{BR}\,Q_t^i(s,\vec{\pi}^{-i}(s))$, i.e.,

$$\mathrm{BR}\,Q_t^i(s,\vec{\pi}^{-i}(s)) = \max_{\pi^i(s)\in PD(A^i)} Q_t^i(s,\pi^i(s),\vec{\pi}^{-i}(s)) \quad (14)$$

$$V^{i*}(s) = \mathrm{BR}\,Q_t^i(s,\vec{\pi}^{-i}(s)) \quad (15)$$

From Eq. (14), we can see that in this algorithm, agent i also needs to learn other agent's actual strategy $\vec{\pi}^{-i}(s)$. The agent i can use her past experiences to update her belief on other agents' strategy $\hat{\vec{\pi}}^{-i}(s)$ in state s in the way that is similar as in fictitious play. When at time t, the state is s and agent j ($j\neq i$) takes action a^j_t. Denote $k^j(s,a^j)$ to be how many times the tuple (s,a^j) has been visited and update $k^j(s,a^j)$ with

$$k^j_{t+1}(s,a^j) \leftarrow k^j_t(s,a^j) + \begin{cases}1 \text{ if } a^j_t = a^j\\ 0 \text{ if } a^j_t \neq a^j\end{cases} \quad (16)$$

So, we have:

$$\hat{\pi}^j_{t+1}(s,a^j) = k^j_{t+1}(s,a^j) \Big/ \sum_{\vec{a}^j\in A^j} k^j_{t+1}(s,\vec{a}^j) \quad (17)$$

In online learning, the agent's current best action may not be the actual best. Therefore the agent has to explore the other actions. By exploiting actions that have been proven to be successful, it is possible to perform well; but by exploring alternative actions, it is possible to perform even better. One popular method to handle the trade-off between exploration and exploitation is ε -greedy: if a^{i*} is the current (time t) optimal action and s is the current state, with probability $1-\varepsilon$, exploit—take action a^{i*}, but with probability ε, explore — choose an action at random. Typically, ε is decayed over time (e.g., $\varepsilon \propto 1/t$). Such an exploration-exploitation method satisfies the GLIE (Greedy in the Limit with Infinite Exploration) property defined by Singh et al. in [8]. a^{i*} in this method is given by the definition of best response:

$$a^{i*} = \arg\max_{a^i\in A^i} Q_t^i(s,a^i,\vec{\pi}^{-i}(s)) \quad (18)$$

4.2 Algorithm description

Let the learning agent be indexed by i, the algorithm called Multi-agent Q-learning with BR strategy can be described in Table 1.

MULTIAGENTQ-LEARNING (MarkovGame, γ, α)

Inputs discount factor γ

learning rate α

Output Q-value functions $Q^i(s,\vec{a})$

Initialize $Q^i(s,\vec{a})=0$ for all $(s,\vec{a})$

1. Initialize $\vec{a}=(a^1,\cdots,a^n)$, s, and belief on the other agents' actual strategies $\hat{\vec{\pi}}^{-i}(s)$ for all s. Take action a^i.
2. Observe rewards r^i_t, next state s', and actions taken by all agent $\vec{a}=(a^1,\cdots,a^n)$.
3. Update $\hat{\vec{\pi}}^{-i}(s)$ with Eq. (16) (17).
4. Compute $\mathrm{BR}\,Q_t^i(s,\vec{\pi}^{-i}(s))$ with Eq. (14).
5. Update $Q^i_{t+1}(s,\vec{a})$ $\leftarrow(1-\alpha_t)Q^i_t(s,\vec{a}) + \alpha[\,r^i_t + \gamma\,\mathrm{BR}\,Q^i_t(s,\vec{\pi}^{-i}(s))\,]$.
6. Choose action a^{i*} (a^{i*} is given by Eq. (18)) with probability $1-\varepsilon$; choose random action with probability ε.
7. Decay α according to schedule.
8. Return to Step 2.

Table 1: Multi-agent Q-learning with BR strategy

5 Other solutions

In this section, our new algorithm is compared with other learning algorithms in the literature. The relationships between these algorithms are also pointed out.

The learning algorithms in MASs differ on what is the main reason for them to introduce learning in Markov games. Algorithms with the different research background will have different consideration. For the field of reinforcement learning, learning is the way to overcome the lack of the information about state transition probabilities and reward functions. For the field of game theory, learning is the way to overcome the defects of equilibrium especially Nash equilibrium. Standard game theory assumes that the rationality and rewards of all the agents are common knowledge: each agent is then able to compute the set of possible equilibria, and if there is a unique equilibrium, choose an action to follow equilibrium. It can't explain when there are multiple equilibria, how agents come to expect the same equilibrium. And it makes strong assumption about the other agents' rationality, which may

not be satisfied in MAS consisted of heterogeneous agents. Here we present two widely used algorithms that come from two fields respectively.

5.1 Fictitious Play

A widely studied model of learning in game theory is the process of fictitious play [9]. In it agents assume that their opponents are playing a fixed strategy. By counting the frequency of other agents playing certain actions with Eq. (16), an agent can figure out the approximate probabilities by which other agents take certain actions (see Eq. (17)). These probabilities are the learning agent's beliefs of other agents' actual strategies. In each round, the agent chooses its action as a best response to its belief of other agents' strategies to maximize the expected reward.

Our algorithm use the same opponent modeling approach and also adopt best response strategy, so fictitious play and our algorithm are both opponent-dependent, and still work when there are multiple equilibria or opponent has rational limitation. There are also some differences between two algorithms. First, in our algorithm best response strategy is calculated based on Q-value not reward directly. In fictitious play, best-response strategy is defined for one period, while our strategy covers the whole course of the game. Second, in our algorithm the probability of choosing an action is not directly associated with the learning algorithm itself. An agent can randomly choose an action or can choose an action based on exploration vs. exploitation consideration. So our algorithm can take into account the dynamic of Markov game when making decision.

5.2 Nash Q-learning

Hu and Wellman [10] consider the learning problem in MAS from the view of reinforcement learning. They first define Nash equilibrium policies as the solution of Markov game. The problem remained is how to achieve equilibrium policy for learning agent who lack the information about reward. Naturally, approach to converge is Multi-agent Q-learning: each agent selects the equilibrium policy according to the current value function. The value function is then updated based on the actual rewards of following these policies.

Hu and Wellman's algorithm (called Nash Q-learning), is very similar to ours. Nash Q-learning also uses Eq. (12) to update Q-table, the difference is the definition of $V^{i*}(s)$. In Nash Q-learning,

$$V^{i*}(s) = Nash_t^i(Q^1, \cdots, Q^n) \tag{19}$$

Consider the same stage NG at time t. Let $Q_t^i(s,\vec{a})$ be the reward of agent i for actions profile $\vec{a}$ and state s. $Nash_t^i(Q^1, \cdots, Q^n)$ denotes the i th agent's reward according to Nash equilibrium in this game.

Comparing with our algorithm, Nash Q-learning have the following limitations: First, both algorithms require observations of other agents' actions, but Nash Q-learning also requires observations of their individual rewards. In network context, reward functions are often in the form of utility. Utility is the private information of individuals and difficult to be revealed to others. Our algorithm bypassed this difficulty. Second, while our algorithm only need maintain one Q-table Q^i, Nash Q-learning need maintain n Q-tables $(Q^1, \cdots, Q^n)$ to calculate Nash equilibrium, which is exponential in the number of agents. So this algorithm is not scalable, while scalability is the basic request for algorithm in most network context. Third, Nash Q-learning is opponent-independent, learning agent converges to Nash equilibrium strategy no matter what actual strategies other agent adopted are. When one agent is not perfectly rational, learning agent's strategy is also not optimal. And this algorithm can not work when there are multiple equilibria.

Greenwald and Hall [12] proposed correlated Q-Learning algorithm, by defining $V^{i*}(s)$ as

$$V^{i*}(s) = CE_t^i(Q^1, \cdots, Q^n) \tag{20}$$

where $CE_t^i(Q^1, \cdots, Q^n)$ denotes the ith agent's reward according to some correlated equilibria with the same stage NG at time t in Nash Q-learning. In this algorithm, agents must collaborate to jointly learn equilibrium policies, which overcomes the problem of multiple equilibria, but makes even stronger assumption about other agents' rationality.

In summary, by synthesizing fictitious play's best response strategy and Nash Q-learning's multi-agent Q-learning, our algorithm consistently shows best performance form any field's point of view.

6 Conclusions

In network contexts, multi-agent systems can be viewed as Markov games where the artificial agents are self-interested (sometimes limited-rational) reward- maximizers with incomplete information about the other agents in the system. This paper investigates the problem of learning in this situation. We propose a new multi-agent learning algorithm in which learning agent selects the best response policy according to the current value function, and the value function is then updated based on the actual rewards of following these policies. Our approach expands previous research on multi-agent learning in both game theory and reinforcement learning communities and provides an applicable algorithm that can be used in network context.

There are a number of important areas remained for future work in this paper, such as some kinds of multi-step backup algorithms for Markov games. The case when there exist learning agents with non-stationary strategy, and algorithms that can handle less information, for example, does not require observing the other agents' actions. All these are necessary to applying multi-agent learning to large problems.

Acknowledgments

This research was sponsored by the China Hubei Provincial Foundation for Natural Science under Grants No. 99J041 and 2001ABB104. The authors gratefully acknowledge this support.

References

[1] G. Weiß and S. Sen, editors, Adaptation and Learning in Multiagent Systems, Springer, 1996.

[2] P. Stone and M. Veloso, Multiagent Systems: A Survey from a Machine Learning Perspective, Autonomous Robots, 8(3), pp.345-383, July 2000.

[3] C. Zhang, J. Wei, An economic model for QoS guarantee in the Internet, Proceedings of SPIE Volume: 4556, pp. 116-121, Sept. 2001.

[4] D. Fudenberg and J. Tirole, Game Theory, The MIT Press, 1991.

[5] M. L. Littman, Markov games as a framework for multi-agent reinforcement learning, Proceedings of the Eleventh International Conference on Machine Learning, pp. 157 - 163, Morgan Kaufman, 1994.

[6] A. Greenwald, Learning to Play Network Games, New York University Dissertation, 1999.

[7] C. J. C. H. Watkins and P. Dayan, Q-learning, Machine Learning, 3, pp.279-292, 1992.

[8] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvari, Convergence results for single-step on-policy reinforcement learning algorithms, Machine Learning, 38(3), pp.287-308, 2000.

[9] D. Fudenberg and D. Levine, The Theory of Learning in Games, The MIT Press, 1998.

[10] J. Hu and M. Wellman, Multiagent reinforcement learning: Theoretical framework and an algorithm, Proceedings of the Fifteenth International Conference on Machine Learning, pp.242-250, 1998.

[11] A. Greenwald and K. Hall, Correlated Q-Learning, Collaborative Learning Agents, K. Tumer and P. Stone Eds. Madison, WI: AAAI Press, 2002, pp. 84-89.

[12] M. Bowling, Rational and convergent learning in stochastic games, Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pp.1021-1026, August 2001.